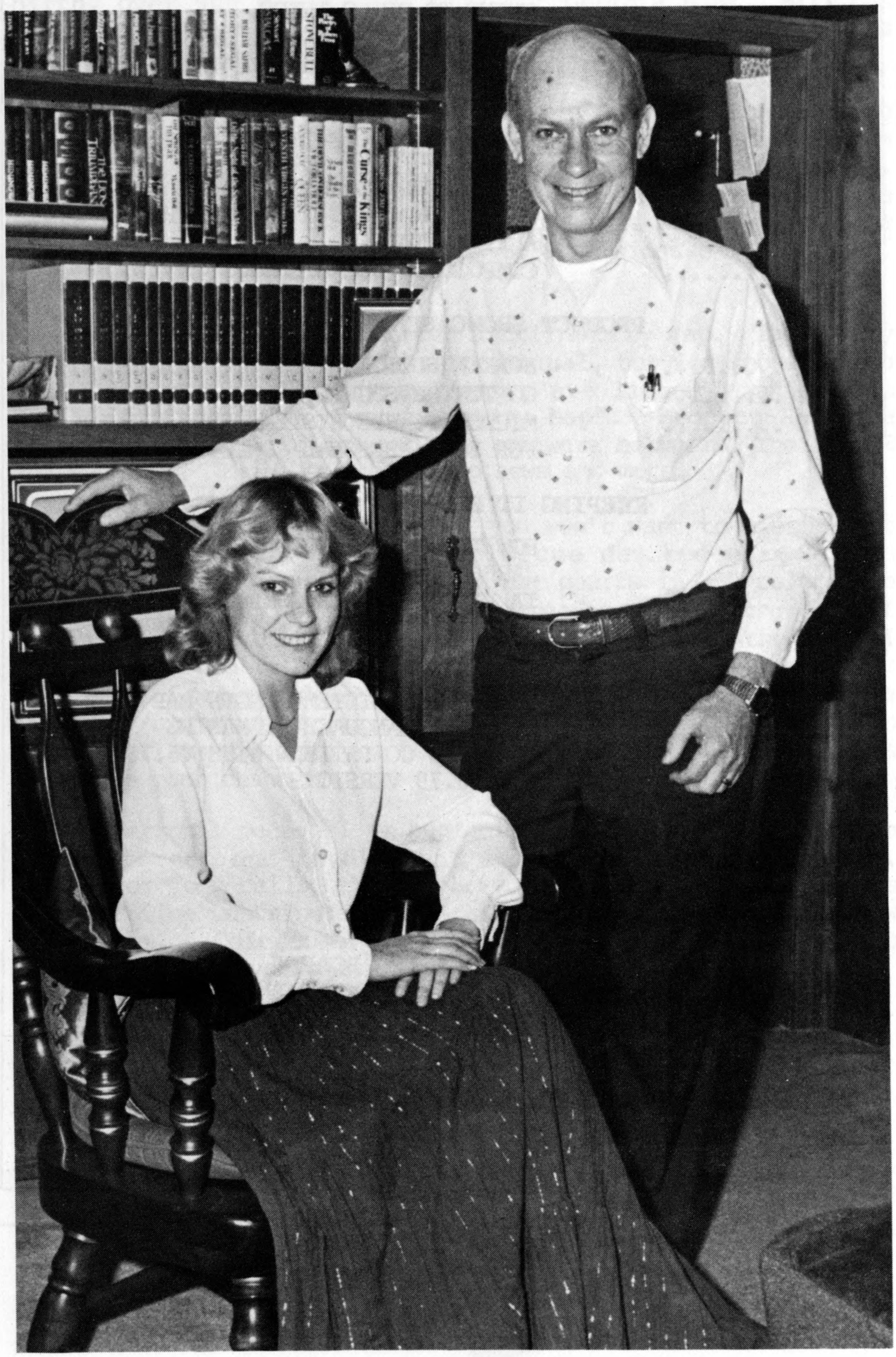
colorcue

a publication for Compucolor users · v.3, #1 · dec / jan. 1980 · $1



inside

colorcue

menu

contributing to the success of this issue

USER INPUT

BEN BARLOW
BILL GREENE
& FAMILY
DENNIS MARTIN

TECHNICAL ADVICE

KNOX PANNILL
BRUCE WILLIAMS

ART DIRECTOR

DEBBIE FELDMAN

EDITOR

CATHY ABRAMSON

—member international association of business communicators—

COLORCUE is published monthly for the users of the Compucolor II personal computer by Compucolor Corporation, P.O. Box 569, Norcross, Georgia 30071. Compucolor is a wholly owned subsidiary of Intelligent Systems Corp. Address all COLORCUE correspondence to our editorial office located at 100 Northcreek, Suite #250, 3715 Northside Parkway, N.W., Atlanta, Georgia 30327, (404) 261-3003. Subscriptions to COLORCUE are $12.00 per year in the U.S., Canada, and Mexico, and $24.00 elsewhere.

editor's letter

This issue will be a combined December/January issue which will bring our publication back on schedule at last. It is full of interesting articles, many submitted by you! We have an interview with Bill Greene, who uses the Compucolor in a very special way -- to communicate with his daughter, deaf from birth, over 900 miles away from her Georgia home. Bill has written a companion article which describes how he interfaces his Compucolor and Teletype.

Ben Barlow, of the Rochester Users Group, has sent us an article with information that many of you have requested -- how to use the RS-232 port to interface with a timesharing system. For you assembly language fanatics, our assembly language article this month has a routine which makes assembly language programs compatible for V6.78 and V8.79 versions of the system.

Many of you have requested a CCII memory map. The memory map is located in the **Maintenance Manual**, but for those of you who don't own it, memory addresses are listed in an article entitled "Compucolor System Memory Map". Product Showcase reviews the Screen Editor and two software networks, The SOURCE and MicroNET. There is user group news and more.

February's issue will be one you won't want to miss -- our GRAPHICS SPECIAL! A whole **Colorcue** devoted to graphics including a program which makes bar charts in 3-D color, one which rotates a cube and more! I need your help though. If this issue is to be really exciting, I need some of the special applications you've developed. Send me color graphics programs for this issue and help the cause. I will be accepting articles right up until deadline -- February 8th. The March issue will be devoted to hardware, so all you hardware buffs -- here's your chance!

A personal note: I am beginning to see a real increase in your interest and involvement and that's very gratifying. Thanks for your calls, thanks for your letters, and many thanks for your fine contributions. I am really excited about the new direction of **Colorcue**. It is your interest that makes my job as editor so enjoyable and rewarding.

Cathy Abramson

=

All of us know that the Compucolor readily lends itself to a wide variety of applications. Compucolor buyers are generally more sophisticated, the ones who want to get more from their purchase than a fun game -- so pat yourself on the back! The Compucolor's generally being used for more than one purpose, in many cases this includes some sophisticated work-related activities, serious personal applications, and, of course, for personal enjoyment as well.

Bill Greene and his family are no exception in this respect. They have found a wide variety of ways to make the Compucolor one of their family. The Greene family is a close one and exceptional in some respects. Of Bill's three children, two are deaf from birth.

Kathy Greene, 23, studies Medical Technology at the National Technical Institute for the Deaf in Rochester, New York. When she graduates from NTID this May, Kathy will continue her studies at Galaudet College in Washington. Like most of the deaf these days, Kathy reads lips as well as signs. She also speaks quite well. She and her brother Tracy (also deaf) have both attended and graduated from public schools since very early childhood. Bill uses his Compucolor to keep in touch with Kathy over the miles.

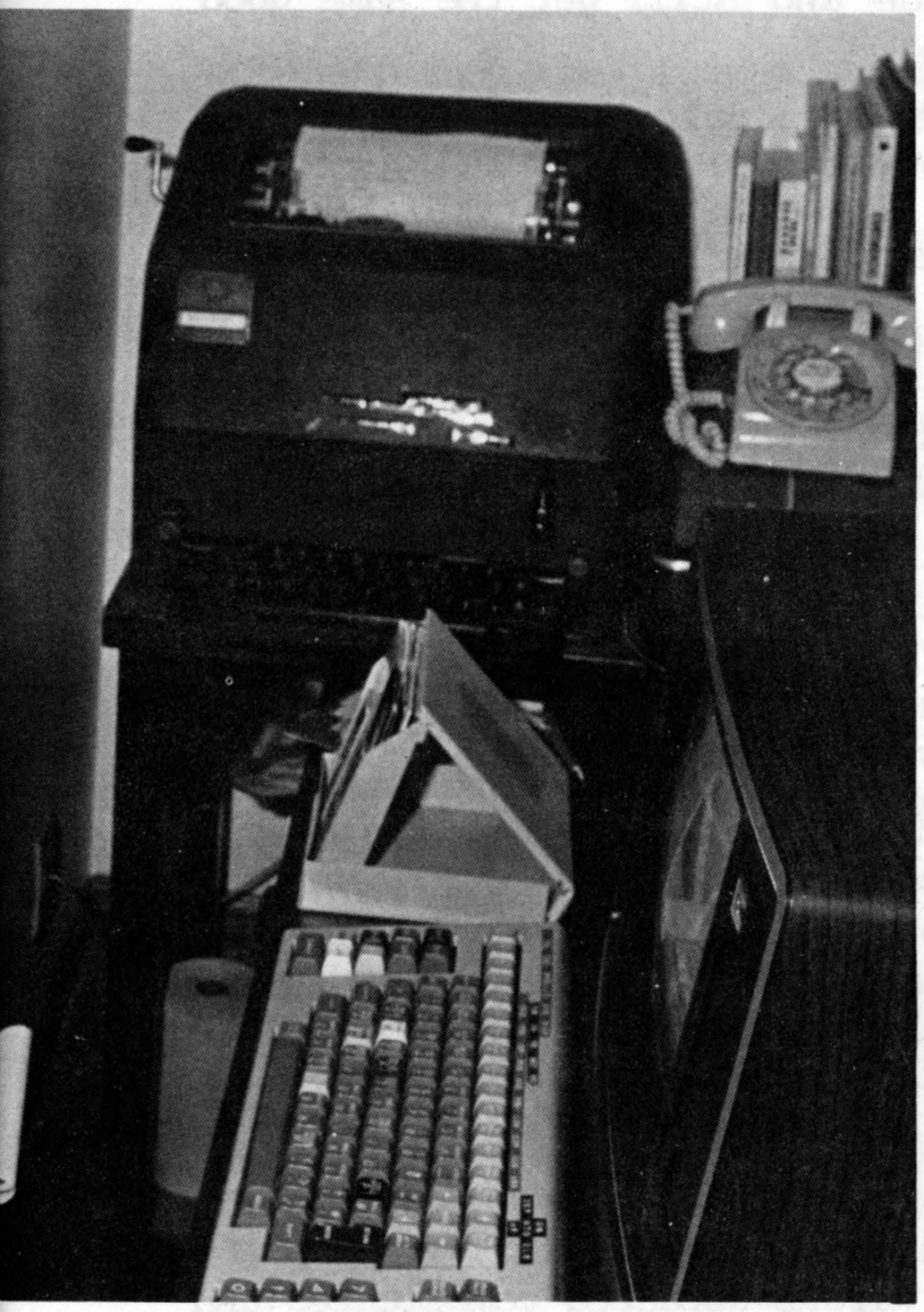
Compucolor II: The next best thing to being there.

"I had been using the Teletype for ten or fifteen years. That is more or less the standard for deaf communication right now, that Model 15 Teletype. The main advantage to using the Compucolor," Bill explained, "is that I'm not an accomplished typist at all (although Kathy is!), so any long conversation gets to be expensive. I can preprogram a message on the screen and send it out at 60 words per minute. Not only that, while she's typing back to me and I'm receiving on the Teletype, I'm preparing an answer on the Compucolor. When she gets through, I push a button and it sends my answer back to her. It gives me some time to type while she's communicating with me." By interfacing his Compucolor with the Teletype machine, the Greenes are able to speed messages to their daughter.

And Bill has thought of some other interesting ways to help the deaf by using the Compucolor. "I've talked with some of the deaf and people working with the deaf about the possibility of using it as sort of a newsletter...Atlanta has one but its not in computer language, it's in the Baudot code that the Teletype software operates on." Galaudet College for the Deaf in Washington, D.C. prepares a daily local news newsletter. The deaf can call a special phone number to get this newsletter transmitted to their Teletype. Many call long distance each day. Bill hopes in the future to be able to make one phone call to Washington and "get that letter, save it on computer and make it available." The deaf could save long distance charges and call locally to get the news of the day.

Bill also uses his Compucolor for some work-related projects. Bill's Compucolor keeps track of chemical inventories at work. He has written a program which calculates the amount of chemicals being stored in tanks of various shapes and sizes. By entering the circumference and height of the tanks, Bill can get a chart which tells him how many gallons of chemicals he has stored. By entering the specific gravity of a particular product he can also get the pounds per gallon for any given number of inches being held in those tanks. Bill's charts are being used at work when they take chemical inventories.

One advantage to having worked with



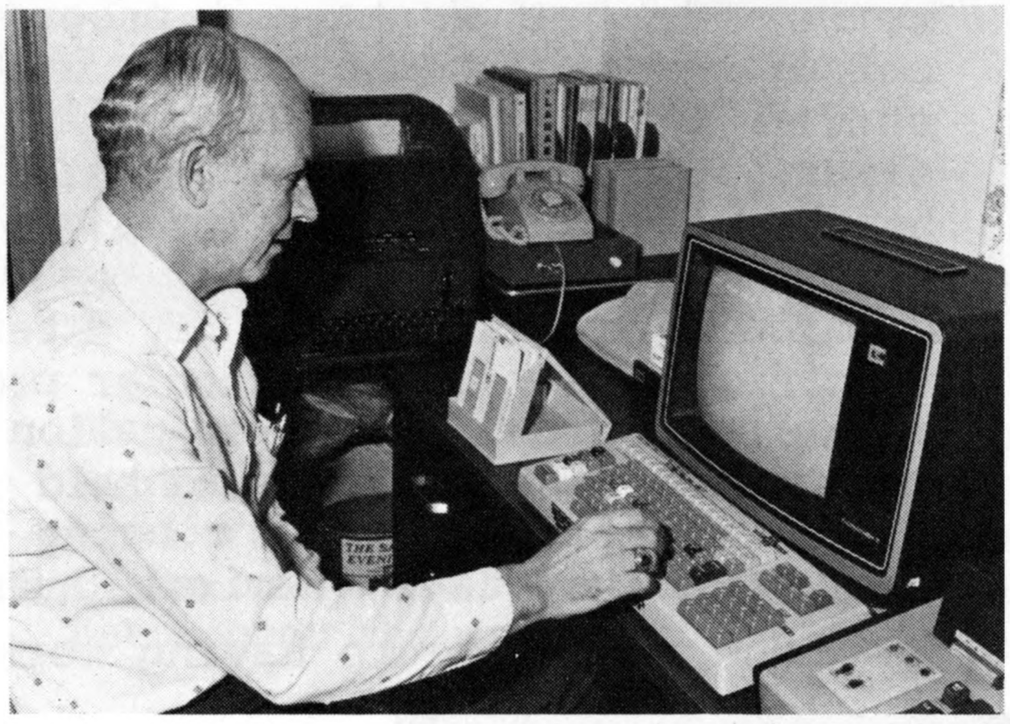
Upper Left: Kathy is currently Miss Deaf Georgia and will compete in the Miss Deaf America contest in Cincinnati, Ohio during the last week of June.

Center: Bill uses his Compucolor to keep in touch with Kathy over the miles.

Right: The Greenes
Left to right, Top: Tracy, Bill, Cheyanne, Bottom: Melissa, Kathy.

the Compucolor, Bill confided, was that now he **knew** when his computer department was making excuses. When they say a job can't be done on computer, they had better be right!

What's in the future for Bill? Well, as it turns out, the future is in the stars! "I'm an amateur astronomer on the side too, and I've already written a program that does sidereal time conversions which prints out local time and siderial time simultaneously on the screen." Bill is very enthusiastic about his astronomy. Bill's star locator "is just a hunting guide, I guess you might say. If you want to look at a certain object, it will tell you whether or not it will be visible at the time you want to try to see it. You can enter the name of an object or a star's number by itself, and it will tell you where the star is located at that particular time--so many degrees above the horizon and so on." His program will not only give the object's direction but will also provide the actual degrees needed to aim the telescope.



"Right now, I've only got about 100 objects listed. I want to do all of the bright stars and all of the more famous double stars on a separate file and then I will have it ready to submit." Bill has his program tied down to this home right now. He hopes to modify it in order to allow others to enter their own latitude and longitude and still get the same good results.

Bill is very proud of his Compucolor. He bought it in December 1978 and says that buying the Compucolor was one of the best decisions he's made. "I wouldn't dare take it back!" he emphasized, "I'm very glad that I made that decision."

Bill Greene has demonstrated quite a bit of ingenuity in his use of the Compucolor. See his article on interfacing the Compucolor and the Teletype which follows. We are happy to print your special applications. Send them to me care of our editorial office.

=

ADVENTURE WITH ASSEMBLER: INTERFACING THE COMPUCOLOR WITH THE TELETYPE

By Bill Greene
of Byron, Georgia

When I purchased my Compucolor II Model 4 in December of 1978, I was already the proud owner of an old Model 15 Teletype that was being used for telephone communication with my two deaf children. It soon occurred to me that I was only about two feet away from hard copy for my computer, cheap hard copy at that. Essentially, only two main problems had to be overcome. The CCII spoke only ASCII and the TTY only understood Baudot, the old 5-level communications code. The CCII has an RS-232C output while the TTY operates with a 60mA current loop.

A study of available literature revealed that both software and hardware conversions of ASCII to Baudot were possible. The hardware conversions appeared complicated and

costly to me, so I elected to go the software route. RS-232C to 60mA current loop, by necessity, has to be done in hardware, but is relatively simple. Actually, only 6 components are necessary including an optical isolator to protect the computer and a TV power transistor to handle the 100VDC 60mA load of the TTY. Fortunately, I did not have to build the power supply for the TTY. It was already there in the modem that I use for telephone communications with the deaf. In use, the RS-232 signal from the CCII is converted to TTL. The TTL signal controls the optical isolator, which in turn opens and closes the power transistor in series with the current loop.

With the hardware problem solved, I ran into an immediate snag with the baud rate. The TTY operates at approximately 45 baud. The RS-232C output of the CCII is normally 110 baud minimum. That meant that the baud rate had to be controlled in software. Turning to the programming manual for help, the section on the TMS 5501 I/O Controller proved to be invaluable. I learned that issuing discrete commands OUT4,0 and OUT4,2 turns on or turns off the RSS-232C output. I eventually used OUT4,8 and OUT4,10 which are the same commands with all interrupts enabled. Issuing these commands at the proper rate in a loop results in software control of the baud rate.

Next, I had to find a source of ASCII characters for input to the proposed ASCII-Baudot conversion program. Initially, I selected screen refresh memory as a convenient source. With text on the screen, every other screen memory byte is ASCII with the color control characters sandwiched in between.

From what I had learned of what others have done, it was apparent that the program should be in machine language, either in PROM or loaded and protected in high memory. At this time I ordered the Assembler and Text Editor diskettes.

While waiting for the Assembler and wanting to get a quick test for my hardware, I wrote the conversion program in BASIC. It was quite a thrill to see the TTY printer respond with garbage the first time I called it from the computer. A few adjustments of the baud rate timing loop resulted in beautiful, clear, hard copy at the agonizing rate of 30 wpm. Each character was being sent at the correct baud rate, but BASIC was taking an equal amount of time with the conversion between characters. However, I was happy, for slow copy was certainly a vast improvement over no copy.

With the Assembler and Text Editor in hand, I made what was perhaps the best decision of the whole project. I decided to write a short assembly language program of the output routine only, and call it from BASIC. This gave me valuable experience in assembly language programming with only a few lines to debug.

With the output routine successfully running in machine language I was still getting 30 wpm because the timing of the output routine had to be the same in either BASIC or machine language. From this point I converted portions of the BASIC program stepwise to assembly language. Each time obtaining an improvement in the character output rate. Finally, with the

whole program in machine language, I obtained the full maximum rate of 60 wpm from the TTY.

PEEKing around memory after using the Text Editor and Assembler, I located the source programs in their respective buffers and added these areas to screen memory as sources of input to my program. This allowed me to list source programs on the printer.

The Assembler output listings, however, were a little more difficult to obtain. I had to partially disassemble the Assembler to find where a ROM utility, CO, 3392H, was called. This utility sends a character in the Accumulator to the screen. I POKEd a call to my TTYPRT program in the call CO locations in order to print the character on the printer. TTYPRT then calls CO, so the character not only gets printed on paper, but on the screen as well.

In conjunction with this project, I wrote a triple function disassembler program in BASIC which outputs each line in decimal, source, and hexadecimal. I found this to be convenient for modifying and debugging machine language programs using BASIC PEEKs and POKEs, which are decimal. The HEX output is useful in saving PRG programs on diskette.

Getting direct hard copy from BASIC and listing BASIC programs was almost by accident. PEEKing around the BASIC scratch pad area, I found a jump to BASOUT, 0033H, in the BASIC output jump vector locations 33286 and 33287. POKEing a jump to TTYPRT in these locations resulted in instant BASIC hard copy for my programs as well as BASIC listings. The TTY prints a 72 character line, so I POKE 33289,72 for full line copy. TTYPRT occupies 302 bytes in high memory, and is protected by initializing BASIC with 16080 bytes available or POKING 32940,208 and 32941,190.

I modified the PRINT.PRG program on the Text Editor diskette by the same procedure used on the Assembler program to get direct hard copy of SRC programs, letters, this article, etc. I have a modification of TTYPRT called KEYTTY that allows use of the CCII keyboard in command mode instead of the TTY keyboard for deaf telephone communications. KEYTTY also allows me to prepare a message in advance to send over the phone at 60 wpm which is considerably faster than I can type.

I can truthfully say that my first experience in assembly language programming was most enjoyable, enlightening, and gratifying. Not only did I gain much insight into the inner manipulation of data in the computer, but I also gained a great deal of respect for the capabilities of the Compucolor II itself.

Because of the uniqueness of this application, I omitted most technical details. However, I would be happy to correspond with anyone interested in Baudot printers or deaf communications.

=

Bill's article should give you all some incentive to go "PEEKing" and "POKEing" around. Bill has also written an IBM Selectric driver program. If anyone needs help with one, Bill says he's willing to share it. If you would like to correspond with Bill, his address is:

EDITOR'S NOTE

Bill Greene
RT3, Box 00-200
Byron, Georgia 31008

=

user group hotline

The following from Bill Shanks of Vicksburg, Ms.:

USER GROUP FOR RADIO HAMS

I want to start a new and different Compucolor Users Group of ham radio operators that will meet on the air on one of the ham bands probably once a week.

Please announce in **Colorcue** that any interested hams should get in touch with me. We would also welcome overseas hams and hams that haven't yet decided to buy their Compucolor.

Bill Shanks, W2GTX
7 Lake Circle Drive
Vicksburg, MS 39180

=

I'd like to remind those of you interested in joining the Education Users Group to contact me as soon as possible. I need your name, address, phone number, as well as level and area of interest.

EDUCATION USERS GROUP OFF TO A FINE START!

I will be getting in touch with those of you who have joined very soon and will also be sending out a membership list. We have had a good response from educators all over the country and we'd like to encourage the rest of you to take advantage of this opportunity while we are still in the formative stage. The more, the merrier! -- Editor.

=

Here is a preliminary list of Compucolor User Groups around the country. I am sure that there are many more. Please contact me with the names of others if you know them. Remember, there is strength in numbers. I will extend your subscription one month for each new user group that you locate.

EDITOR'S NOTE

HEADQUARTERS

COMPUCOLOR USERS GROUP
c/o Compucolor Corp.
Intecolor Drive
225 Technology Park/Atlanta
Norcross, Georgia 30092

EDUCATION USERS GROUP
c/o Compucolor Corp.
Intecolor Drive
225 Technology Park/Atlanta
Norcross, Georgia 30092

ALABAMA

COMPUCOLOR USERS GROUP
Contact: Eike Mueller
12117 Comanche Trail, S.E.
Huntsville, Alabama 35803
(205) 883-7614

Just formed, no dues or publications.

CALIFORNIA

COMPUCOLOR USERS GROUPS
Contact: Mark Nehamkin, Co-President
c/o Intersell
540 Weddell Drive
Suite 9
Sunnyvale, California 94086
(408) 734-5201

$10 membership fee, 30-40 members. Membership entitles you to a copy of the library. Each member must submit one program to the library. No publication.

COMPUCOLOR/INTECOLOR USER GROUP
Contact: Stan Pro
S.P. Electronics
5250 Van Nuys Boulevard
Van Nuys, California 91401
(213) 788-8850

Membership - $30 per year. Each member will receive four bulletins (about 30 pages) packed with technical and programming data. Back issues available. They have a large library.

GOTO GROUP
Contact: Tommy W. Schenck
c/o Tom & Bobbie of Newberrys
1136 Fulton Mall
Fresno, California 93721

No dues, no publications.

NORTHERN CALIFORNIA USERS GROUP
Contact: Barry L. Parr
c/o Creative Publications
P.O. Box 10328
Palo Alto, California 94303
(415) 968-3977

SAN DIEGO COUNTY COMPUCOLOR II USERS' GROUP
Contact: Jim Helms
c/o Color Computing
4888 Ronson Court
Suite H
San Diego, California 92111
(714) 565-7283

Publication: **NEWSLETTER**, free with program submission.
242 programs in user group library (30 disks)

COLORADO

COMPUCOLOR II USERS GROUP
Contact: Chris Carson
1850 S. Truckee Way
Aurora, Colorado 80012
(303) 755-1709 (home)
(303) 292-6670 (office)

The group meets every 2nd and 4th Saturday at Lowry Air Force Base, where they have access to classroom facilities and a complete electronics lab. They are presently in the organizational stage and have no membership fees or publications. Any CCII user or other interested parties are invited to attend.

FLORIDA

ROBINSON HIGH COMPUTER CLUB
Contact: Mrs. Byman
6311 S. Lois Avenue
Tampa, Florida 33616
(813) 835-1211

No dues, no publications.

GEORGIA

COMPUCOLOR USERS GROUP
Contact: Tully Johnstone
4044 Payton Woods Drive
Tucker, Georgia 30084
(404) 493-1657 (home)

The group meets on the 1st Wednesday of every month. There is a $5.00 annual membership fee. No publication yet. Approximately 75 members.

MISSISSIPPI

HAM RADIO USERS GROUPS
Contact: Bill Shanks, W2GTX
7 Lake Circle Drive
Vicksburg, Mississippi 39180

This group is just forming.

NEW YORK

COMPUCOLOR USERS' GROUP OF ROCHESTER, NEW YORK
Contact: Ben Barlow
161 Brookside Drive
Rochester, N.Y. 14618
(716) 385-2969

Publication: **DATA CHIP** monthly, $10.

COMPUCOLOR USERS' GROUP OF ROCKLAND COUNTY
Contact: Ronnie Schnell
17 Eckerson Lane
Spring Valley, N.Y. 10977
(914) 352-8069 (after 6)

Publication: **COMPUSOURCE** $6.00, discounts on Compucolor materials.

=

INTRODUCING THE SCREEN EDITOR

Here's an easy way to edit and create source files or text on screen--with the Compucolor Screen Editor. The difference between the Screen Editor and FREDI is that FREDI edits in BASIC while the Screen Editor edits ASCII.

The Screen Editor requires 16K of memory and the deluxe keyboard. Using PLOT F0 - F15 keys, the Cursor Controls, and the Color Selection Cluster the Screen Editor performs four special functions: disk handling, cursor control, editing, and search.

Disk Controls

The disk handlers include seven keys -- some of which you are already familiar with. Once text and the Editor have been called up, these keys will manipulate the file for you. There are four ways of exiting the file -- two which will save it and two which will not. EXIT (F1) saves the file and returns you to FCS. RECYCLE (F2) saves the file and reopens it for further editing and review. QUIT (F0) returns you to the FCS mode without saving your file, and CPU RESET, of course, returns you to the CRT mode without saving your file.

In addition, there are three other disk handlers. READ PAGE (F4) displays everything presently located in the buffer. WRITE PAGE (F5) writes the current page located in the buffer to the disk. NEXT PAGE (F3) performs a combined read/write, writing the former page to the disk and displaying your next page of text on the screen.

The Colorful Cursor

The colorful and intelligent cursor performs several duties for the Screen Editor. A cyan cursor shows where tabs are located. A blue cursor means that text exists between the cursor and the next carriage return. A magenta cursor comes at the end of a line, and, finally, the white cursor tells you when you have reached the end of your file.

The cursor can be moved in normal fashion through the use of the gray cursor control keys -- up, down, left, right, and home. It can also be moved by using the color pad on the left. Left one character (yellow), right one character (white), down one line (green), up one line (cyan), down fast (1/4 screen, 6 lines - red), up fast (magenta), down screen (full screen, 24 lines - black), up screen (dark blue). TOP PAGE (F6) moves the cursor to the top of the page and BOTTOM PAGE (F7) to the end of the page. These additional cursor controls make editing rapid and easy.

Editing Controls

There are four editing controls which allow you to delete text. DELETE CHARACTER (F11) deletes the character located under the cursor. DELETE CHARACTER (the brown key, top right) and ERASE LINE will both delete a character appearing before the cursor. And DELETE LINE (F12) deletes all characters from the cursor to the end of the line (marked by a carriage return).

Search Controls

The Screen Editor also makes searching for strings easy. F8-10 allows you to search for a specific string in REVERSE (to the top of the buffer), FORWARD (to the end of the buffer), and END (to the end of the file).

Some interesting notes about the Screen Editor. INSERT CHAR is not valid for the Screen Editor. It will, if hit, appear in the search string. Valid Control characters are ERASE PAGE, Control L (form feed, new page), Control I (tab), and Control J (line feed).

Because the Screen Editor is an ASCII editor, lower case characters should not ordinarily be used unless you have the **lower case** option. Lower case characters will be displayed as special characters on the screen if used and will not print correctly unless you have a printer which accepts lower case characters.

The Screen Editor, like FREDI, is a powerful editing tool, and can make your programming time much more enjoyable and efficient.

=

EDITOR'S NOTE

As a service to our readers, Compucolor Corporation publishes information sent to us by other manufacturers of products, techniques, or scientific and technological developments which can be used in conjunction with the Compucolor computer. However, in many cases, Compucolor Corporation has not had the opportunity to thoroughly examine or test these products. Therefore, we neither endorse nor assume any responsibility or liability for the proper functioning of products not directly manufactured by the Compucolor Corporation. We hope you will find these products of interest, but we encourage you to investigate carefully before you buy.

=

NETWORKING

Timesharing networks are springing up all over the country and are becoming more sophisticated each day, adding a whole new dimension to the home computer. Compucolor II is an ideal terminal to use for timesharing purposes.

GO TO THE SOURCE

The SOURCE network is available in most major cities through a local phone number (no long distance charges). The initial membership fee is $100. There is a monthly surcharge of $5, and a $15 per hour charge for use during the day. At night the cost drops to only $2.75 per hour making it very reasonable. Some of the SOURCE's feaures are:

1. The ability to send and receive mail from others,
2. The ability to talk to another user,
3. UPI news available,
4. Numerous programs available,
5. Large storage available,
6. Shopping through the network (for example, enter your Master Charge number and have merchandise sent to you).

Future capabilities include the ability to make reservations, bank, and so on--the sky's the limit. And the SOURCE exists right now! For further information about the SOURCE, contact:

Telecomputing Corporation of America
1616 Anderson Road
McLean, Virginia 221001
(703) 821-6660

MICRONET: BIG SYSTEM PERFORMANCE FOR YOUR PERSONAL COMPUTER

MicroNET is another remote, on-line service available in 27 major metropolitan areas through local telephone numbers. MicroNET is owned by CompuServe and offers the following services:

1. Ability to communicate with other users,
2. Ability to buy and sell software through the network,
3. Availability of personal programs
4. Availability of business applications
5. Availability of educational aids
6. Easy-to-use programming languages
7. Advanced programming and diagnostic tools
8. Entertaining games

plus up to 128K bytes of on-line file storage.

MicroNET hours are 6 p.m. to 5 a.m., Monday through Friday, all day on weekends and most holidays. There is a one-time sign-on charge of $9.00 and a $5.00 per hour rate, all of which may be billed to Master Charge or Visa. 153 other cities may access the system for an additional $4.00 per hour charge.

Send inquiries to:

CompuServe
Personal Computing Division
5000 Arlington Centre Boulevard
Columbus, Ohio 43220

=

keeping it simple

HOW TO POKE WITHOUT GETTING JABBED

By Dennis Martin
of Omaha, Nebraska

Have you ever had a problem with BASIC listings that have one or more line numbers out of sequence? You might wonder how it can happen, but it is easier than you think. It can be as simple as striking ESC instead of a "1" when typing a line, or by having faulty logic in determining a POKE address.

Do not despair, the problem is easily correctable. Just follow these easy instructions, and you can't go wrong. (You can't do any worse than you have already!)

ACTUAL EXAMPLE:

```
1300 IF P>9 THEN 1340
1310 PRINT P;" ";
1320 RETURN
1340 IF P>15 THEN GOSUB 1400
838 PP$=MID$ (STR$ (P), 2, 2)
1360 PRINT PP$;" ";
1370 RETURN
```

In the above example notice that line 838 is not in its proper place in the listing. Also, the computer will not recognize 838 as a valid line number. (Executing a GOTO 838 will result in a US ERROR.)

Now comes the heart of the problem; how to get rid of the line without destroying your program. The first step is to get a pencil, and paper. Now we have to find the actual memory addresses for the bad statement in RAM. This is best done by inserting the following statement in your program:

```
1325 REM ####################
```

This statement must be entered TWO statements before the error! If you had tried to enter this as a statement number above 1340, the computer would have put it AFTER the error. Once you have the statement typed in, list the lines involved to make sure that the REM statement is before the error. If everything has been done correctly, you should now have a REM statement between lines 1320 and 1340.

To do this step requires that you use your own judgement to gauge the approximate location in memory. The easiest way is to pick an address that you believe to be below the address of the statement you are looking for. When you have picked a starting address, enter the following statement in immediate mode:

```
FOR X=A TO B:PRINT X; PEEK (X),:NEXT
Where A is the STARTING ADDRESS and
B is the ENDING ADDRESS.
```

Hit return, and be ready to hit ATTN BREAK. As the addresses and their contents scroll on the screen, you want to watch for a string of twenty 35's (the ASCII value for #). When this string appears on the screen you are getting very close! This string should be followed by a zero, which is the start of line 1340. You then search for the NEXT zero. This should be the start of the statement in error.

In the example given, this would look like this:

```
37821 0   37822 210      37823 147      37824 70
37825 3   37826 80       37827 80       37828 36
37829 172 37830 197      37831 40       37832 191
37833 40  37834 80       37835 41       37836 44
37837 50  37838 44       37839 50       37840 41
37841 0
```

The byte at 37821 indicates the end of the last line. The bytes at 37822 and 37823 are the low and high order bytes of the address of the next BASIC instruction. (210 = low order and 147 = high order, so the next instruction starts at 210 + (147*256) = 210 + 37632 or 37842 decimal, which is the address immediately after the next zero.) The bytes at 37824 and 37825 are the ones we are interested in. 37824 is the low order byte and 37825 is the high order byte of the line number. (70 = + (3*256), so the line number is 70 + 768 or 838.)

Now that we have the address for the line number, it's simple to change the line number. To change it, you must find a line number greater than the statement preceding the error and less than the statement following the error. (If you have successive line numbers, then you will have to delete a line and then find the new memory addresses. This is a must, because once you delete a line, all memory locations after it are changed.) For our example, we will choose 1350 as the new line number. Convert to high and low order bytes by dividing 1350 by 256. This gives 5 with a remainder of 70. The 70 is the low order byte and 5 is the high order byte. Since 37824 already contains a 70 we don't need to change it. Going back to immediate mode, we simple have to POKE 37825,5.

You are done now. Type LIST and you will find that there is no longer an out of sequence error. If the line in question does not belong there, you can simply delete it. You can also delete the REM that you inserted.

For those of you wondering what all of the other numbers are betwen 37826 and 37840, these are the ASCII values for the characters in the statement. The exceptions are 37829 which is an '=', 37830 which is a 'MID$', and 37832 which is a 'STR$'.

These are three examples of what the CCII uses as tokens or atoms to represent specific commands to BASIC.

=

EDITOR'S NOTE

Dennis has promised me a follow-up article on tokens which I will publish in a future issue. Disclaimer: out of sequence line numbers are not unique to the Compucolor but are a problem people occassionally experience with Microsoft's software.

=

TALKING TO OTHER COMPUTERS

By Ben Barlow of Rochester, N.Y.

One of the nice features of the CCII is its built-in RS-232 port, and the commands which allow you to direct output to the port. With other personal computers, this feature is an expensive option. By using the RS-232 port, you can print (by connecting in a printer of some sort), talk to another CCII locally (by connecting to its RS-232 port), or use the CCII itself as a (pardon the expression) dumb CRT terminal (by connecting to a modem). Several users have mentioned using their CCII's as terminals for timesharing -- connecting through modems and the phone network to large mainframe computers, and simply using the CCII as an I/O device.

That works fairly well using the standard built-in routines. The steps to use are:

- Call the timesharing service and get answer tone from the modem,
- Press reset,
- Press ESC R3 (or another number for a different speed)
- Press ESC M

and you should be connected to the distant computer. Things are not always that simple, however. As soon as the modem is turned on, trash starts coming onto the screen, if extraneous noise enters the modem. Also, a computer that sends lower case characters to you (and you don't have lower case) can cause unreadable conversations. When I sign on to the timesharing service which I use, for example, the computer greets me with the message:

```
rook,-,club,_,rght hump,rook,spade,-,_
```

We limped through conversations like that until I was trained. Another problem was that occasionally a control character would sneak in, and the disk would start to whir, or the screen clear, or begin to type in a strange place. Also, the timesharing computer had a habit of sending DELETE characters after line feed sequences (a carry over from the old days, when the teletypes had to have a few additional milliseconds so they didn't print while the carriage was returning). The DELETE characters, however, instead of being DELETEd, printed as _'s. Clearly it was time for a program.

The following BASIC program helps the CCII to operate as a terminal. It provides the intelligence for it to play "dumb". It screens all characters coming in from the RS-232 port, and

- Changes lower case characters to upper case,
- Deletes DELETEs

- Ignores control characters (those below HEX 20)

Operation of the program is simple. After typing it in and saving it, just:

- RUN
- Press ESC CRT
- Press ESC user (hat)
- Call the distant computer and connect.

As the program operates, it changes the color of the first screen position. This is a debug feature that told me (and you) that the thing was actually working. To omit the feature, add this line to the program:

```
110 POKE 40931,0:POKE 40932,0:POKE 4093,0
```

The Baud rate is set by the program to 300, with one stop bit. To change the speed or number of stop bits, add:

```
115 POKE 40948,x
```

where x is:

speed	one stop bit	two stop bits
110	129	1
150	130	2
300	132	4
1200	136	8
2400	144	16
4800	160	32
9600	192	64

Following the BASIC program is a listing of the assembly language version of the patch. It may help if you need to modify the program.

```
  5 POKE 32940,195:POKE 32941,159
  6 CLEAR 200
 10 DATA 245,123,230,127,254,127,202,235,159,254,10,202,230,
    159,2,54,13
 12 DATA 202,230,159,254,32,218,235,159,254,64,218,230,159,
    230,22,3,50,1
 13 DATA 112,95,241,195,220,57
 20 DATA 241,201,245,62,31,50,227,129,62,132,211,5,62,14,50
    223,1,29,241,201
 30 DATA -1
 50 FOR X= 40900TO 40999
 60 READ A:IF A= -1THEN X= 41000:GOTO 100
 70 POKE X,A
100 NEXT
120 POKE 33221,195:POKE 33222,196:POKE 33223,159
```

```
130 POKE 33215,195:POKE 33216,237:POKE 33217,159
READY
```

Note:

Question, really. Is line 60 as written any different from:

```
60 READ A: IF A = -1 THEN 120
```

? Assuming that temporary space is taken up by FOR:NEXT loops for control, does it get cleared out by escaping from the loop, or does it just accumulate to later cause memory shortfalls? [Answer: Temporary space is taken up, but a RUN, RETURN or a CLEAR clears it up -- Editor.]

```
0000                        ORG 40900
                            ;
                            ;       ROUTINE TO EDIT INPUT
                            ;
9FC4 F5          START:     PUSH    PSW
9FC5 7B                     MOV     A,E
9FC6 E67F                   ANI     7FH
9FC8 FE7F                   CPI     7FH
9FCA CA0000                 JZ      EX2       ;IGN DEL
9FCD FE0A                   CPI     0AH
9FCF CA0000                 JZ      EX1       ;TAKE LF
9FD2 FE0D                   CPI     0DH
9FD4 CA0000                 JZ      EX1       ;TAKE CR
9FD7 FE20                   CPI     20
9FD9 DA0000                 JC      EX2       ;IGN CONT. CHARS
9FDC FE61                   CPI     61H
9FDE DA0000                 JC      EX1       ;TAKE LESS THAN LC A
9FE1 E6DF                   ANI     0DFH      ;EDIT LC
9FE3 320170                 STA     7001H
9FE6 5F          EX1:       MOV     E,A
9FE7 F1                     POP     PSW
9FE8 C3DC39                 JMP     39DCH
9FEB F1          EX2:       POP     PSW
9FEC C9                     RET
                            ;
                            ;       SET UP LINKAGE
                            ;
9FED F5          LINK:      PUSH    PSW
9FEE 3E1F                   MVI     A,31
9FF0 32E381                 STA     81E3H     ;INP. FLAG
9FF3 3E84                   MVI     A,84H
9FF5 D305                   OUT     5         ;BAUD RATE
9FF7 3E0E                   MVI     A,14
9FF9 32DF81                 STA     33247     ;KB FLAG
9FFC F1                     POP     PSW
9FFD C9                     RET
9FFE                        END     START

**********************************
```

```
BR ADDR=9FCB EB 9F
BR ADDR=9FD0 E6 9F
BR ADDR=9FD5 E6 9F
BR ADDR=9FDA EB 9F
BR ADDR=9FDF E6 9F
LABELS:
START          40900 196 159
EX2            40939 235 159
EX1            40934 230 159
LINK           40941 237 159
245 123 230 127 254 127 202 235 159 254 10 202 230 159 254 13
202 230 159 254 32 218 235 159 254 97 218 230 159 230 223 50 1
112 95 241 195 220 57 241 201 245 62 31 50 227 129 62 132 211
5 62 14 50 223 129 241 201 32
```

=

EDITOR'S NOTE

This article was reprinted with the kind permission of Mr. Barlow from **Data Chip,** the newsletter of the Compucolor Users Group in Rochester, N.Y. Mr. Barlow is the editor of that fine publication and I heartily recommend both the users group and the newsletter to those of you in his area. You can contact the Rochester Users Group by writing:

Ben Barlow
161 Brookside Drive
Rochester, N.Y. 14618

=

advanced applications

COMPUCOLOR SYSTEM MEMORY MAP

There are two memory maps for the Compucolor; one for the V6.78 and one for the V8.79 system software. All addresses are hexadecimal.

V6.78 System Memory Map

0000 - 003F	Restart Vectors and Initial Values
0040 - 211B	BASIC ROM
211C - 3FFF	FCS and CRT ROM
4000 - 5FFF	Future Space
6000 - 6FFF	High Speed Screen Refresh RAM
7000 - 7FFF	Low Speed Screen Refresh RAM
8000 - 81FF	System Scratch Pad RAM
8200 - FFFF	User RAM

8000 - 8299	System and BASIC Scratch Pad RAM
829A - FFFF	BASIC User RAM

V8.79 System Memory Map

0000 - 003F	Restart Vectors and Initial Values
0040 - 1F25	FCS and CRT ROM
1F26 - 3FFF	BASIC ROM
4000 - 5FFF	Future Space
6000 - 6FFF	High Speed Screen Refresh RAM
7000 - 7FFF	Low Speed Screen Refresh RAM
8000 - 81FF	System Scratch Pad RAM
8200 - FFFF	User RAM
8000 - 8299	System and BASIC Scratch Pad RAM
829A - FFFF	BASIC User RAM

=

ASSEMBLY LANGUAGE PROGRAMMING 3: MAKING PROGRAMS COMPATIBLE WITH V6.78 AND V8.79 VERSIONS

We have looked at routines for both character output and input. Now we will explore a routine that will make assembly language programs compatible with both V6.78 and V8.79 systems.

There are two versions of Compucolor II system software available. The V6.78 version was released with the first Compucolor II and V8.79 was introduced at the end of Fall 1979. The reason for changing systems had to do with the software modifications involved in changing from 3 phase to 4 phase micro disk drives. (See "Add-On Disk Drives $395" Sept./Oct 1979, v.2,#7,p.6) As a result, some of the key memory locations have changed. Object files (.PRG) may have to be modified in order to run on the other version of system software. This does not effect BASIC programs except where PEEKs, POKEs to the system software or user machine language routines are used. The GET character, Scrolling, and sound patch that many of you are using are **not** effected.

The purpose of the following routine is to allow you to write assembly language programs which will configure themselves at run-time, depending upon the version of system software present in the machine being used.

When writing programs that are to interface with two or more different software systems, ask yourself whether or not the systems contain the same or similar functions and if so, whether the functions are referenced in the same way and are located at the same addresses. In our specific case, the routines and variables are the same; the differences lie in their locations in system memory.

As a general rule, any reference to memory addresses from 0040H to 5FFFH have changed and all other references are the same. Therefore all references to memory outside the changed areas can be handled with "Equate" statements (EQU). The following equates are the same for both versions of system software.

```
KEYTST  EQU     0024H   ; KEYBOARD SCANNER
INPCRT  EQU     81C5H   ; JUMP VECTOR #31
KBDFL   EQU     81DFH   ; HOLDS JUMP VECTOR NUMBER FOR KEYBOARD
```

```
KBCHA   EQU     81FEH   ; CHARACTER FROM KEYBOARD
KBRDY   EQU     81FFH   ; KEYBOARD READY FLAG
```

Another approach must be taken for routines and variables at other locations in the different software versions. The method used here builds a table of jump vectors to the routines and variables in the system. By doing this, only one address per routine or variable must be changed. When a program is loaded and executed for the first time, there must be some way of determining which version of software is present in the system and then the jump vectors must be corrected for that version. You can determine which software version you have by comparing selected memory locations with values known to be there for specified versions of system software.

In our routine, the locations 0001H and 0002H have been chosen for comparison. This decision is somewhat arbitrary. These locations are more likely to accurately reflect the version of software present. Another jump table is made which contains the addresses for the other version to be handled. If more than two system versions are involved, there needs to be a table for each version. In this example, the "main" jump table contains the values for V8.79 and the other table contains those values for V6.78. The "main" table should also contain the appropriate labels for the routines required. This will fix the addresses of the table so that we will know where the changes must be made.

The determination of system software version should be made when the program is first loaded and executed. The routine used to determine the version should only be executed once and should not be executed when the program is re-entered from a path such as <ESC T> or <ESC I>. The routine should also be executed before any references to system functions or variables occur. To make this routine execute only once, begin execution at 8203H for <ESC T> or 9003H for <ESC I>. This allows a jump around the routine to be placed at 8200H-8202H for entry through the <ESC T> vector or to be placed at 9000H-9002H for entry through the <ESC I> vector. In the "load" file (.LDA) this is done by placing the label of the overlay routine immediately following the "END" statement. This sets the starting address for a "load" file. In the case of a "program" file (.PRG), care must be taken when the .PRG file is created using the "SAVE" command in FCS. The format of the "SAVE" command would be:

FCS> SAVE filename (memory range),(starting address)

For a filename of "SPLAT", a loading address of 8200H, a range of 8200H-85FFH, and a starting address of 8203H, the command would be:

FCS> SAVE SPLAT 8200-85FF,8203

Programs that were saved using this method can be found on the "BASIC Editing" disk. It should be noted that the advantage of

using the <ESC I> vector is that BASIC's pointers are preserved so that BASIC can be re-entered. Using <ESC I> to vector to 9000H may, however; destroy a BASIC program that is loaded.

Our routine first tests to determine which version of system software is present. If it is the latest version, then everything is alright and the main program can now be executed. If the earlier version is present, then the jump table for that version must be copied over the main jump table. This is called "overlaying". Once the jump table has been overlayed, the original values are lost and cannot be recovered until the program is reloaded from disk and re-executed. While the determination and overlaying take place, the interrupts are turned off just to be safe. The overlay routine is executed first and only once. Loading the program using the FCS "LOAD" command and then using the <ESC T> vector may produce undesirable results.

There are two approaches as to locating the jump tables. One is to locate them close to the overlay routine so that they can be easily referenced. This method requires that you set aside memory exclusively for the tables. The other approach is to locate the tables at the end of the program, just before the variable storage area. By placing the tables here, with the main jump table first, the variable area can be assigned to overlap the table used for the overlay. This method requires that none of the variables be used or modified until after the overlay has taken place. The main advantage of this approach is that memory can be conserved by using it for more than one purpose. If the latter method is used, the saved object file (.PRG) must contain all of the tables. In our example, the code should be saved through "END678". For the sake of simplicity and/or for systems with plenty of user RAM, the first method is recommended. For those of you who wish to use the second method, the example contains the necessary notes to help you.

Here's the overlay routine, followed by examples of jump tables containing the routines we have just discussed. More routines will be added to the list of those that must be overlayed, later in our series on assembly language.

```
;*************************************************************
;;      ***** START OF PROGRAM *****
;
        ORG     PRORG   ; STARTING ADDRESS OF PROGRAM

START:  JMP     STRT2   ; FOR ENTRY THROUGH [ESC T]
STRT1:

;*************************************************************
;;      ROUTINE FOR OVERLAYING THE SYSTEM JUMP TABLE
;
;       THE FOLLOWING ROUTINE IS FOR SOFTWARE THAT
;       WILL CONFIGURE ITSELF AT RUN-TIME DEPENDING
;       ON THE VERSION OF SYSTEM SOFTWARE THAT IS
;       RESIDENT IN THE MACHINE (V6.78 OR V8.79)
```

```
;
;       *** OVERLAY SYSTEM JUMP TABLE ***
;       (TABLE LOCATED AT END OF PROGRAM)
;
;       IF V8.79 THEN TABLE IS CORRECT
;       IF V6.78 THEN OVERLAY VALUES FOR V6.78

S678    EQU     0376CH  ; V6.78 @ 0001-0002
S879    EQU     001BAH  ; V8.79 @ 0001-0002

INITSJ: DI              ; ** TURN OFF INTERRUPTS **
        LXI     SP,STACK ; SET TEMP STACK
        LXI     H,0001H ; GET ADDRESS FOR TESTING
        MOV     E,M     ; GET LOW BYTE
        INX     H
        MOV     D,M     ; GET HIGH BYTE
        LXI     H,S678  ; GET VALUE FOR V6.78
CMPTYP: ; COMPARE FOR DE=HL <Z>
        MOV     A,D
        CMP     H
        JNZ     STRT2   ; IF NOT V6.78, THEN 'MUST' BE
        MOV     A,E     ;  V8.79
        CMP     L
        JNZ     STRT2
MOVESJ: LXI     D,SJTBL ; ADDRESS OF SYSTEM JUMP TABLE
        LXI     H,OSJ678 ; ADDRESS OF OVERLAY TABLE FOR V6.78
        MVI     B,ENDSJT-SJTBL  ; COUNT OF ELEMENTS IN TABLE
MOVSJ1: ; OVERLAY SYSTEM JUMP TABLE
        MOV     A,M     ; GET ELEMENT FROM OVERLAY TABLE
        STAX    D       ; PLACE IN 'MAIN' JUMP TABLE
        INX     H       ; NEXT SOURCE ADDRESS
        INX     D       ; NEXT DESTINATION ADDRESS
        DCR     B       ; ONE LESS ELEMENT TO TRANSFER
        JNZ     MOVSJ1  ; CONTINUE UNTIL FINISHED
        EI              ; ** ENABLE INTERRUPTS **
STRT2:  ; ** begin program here **
;*****************************************************
;;      *** SYSTEM JUMP TABLE ***

SJTBL:  ; FOR V8.79 SYSTEM SOFTWARE

LO:     JMP     17C8H   ; SEND CHARACTER TO SCREEN
OSTR:   JMP     182AH   ; SEND STRING ENDING WITH 239

ENDSJT: ; END OF SYSTEM JUMP TABLE
;*****************************************************
;;      *** SYSTEM JUMP OVERLAYS ***
SJOVRLY:
OSJ678: ; OVERLAY FOR V6.78 SYSTEM JUMPS

        JMP     3392H   ; LO
        JMP     33F4H   ; OSTR

END678:
;*****************************************************
```

```
        ORG     SJOVRLY         ; USE THIS TO SAVE MEMORY
        ; data area starts here
                :
                :
                :
        DS      100     ; STACK AREA
STACK:
        DS      2

        END     STRT1   ; 'STRT1' IS ADDRESS OF OVERLAY ROUTINE
```

We will discuss FCS file routines, general system routines, using the system flags and handling serial I/O in future issues. If you have any favorite reference materials or other comments and suggestions, please drop us a letter so that we can plan articles which will help you best.

=

input

From Brian Hogan in Auburn, Washington:

My kids soon found out that when they turned on the computer to play their favorite game, many results were totally predictable even though "random" numbers where being used in the program. This happens because the same sequence is generated when one first turns on the computer (see **Programming Manual.**)

Slipping in the following two lines at the first of such a program will give a random beginning to a random number sequence.

```
0 Y=PEEK(33209):IF Y=0 THEN 0
1 Y=RND(-Y)
```

Follow any future calls to the random number sequence with RND(1).

What we've done is to PEEK at the number of seconds elapsed and converted it to a negative number, since RND(neg. no.) gives a different random sequence.

For what it's worth, my own experience with the random sequences on the Compucolor seems to indicate that eventually all sequences get caught up in a loop of 1995 numbers that has .999484 as the largest value and 1.85287E-04 as the smallest value. (Before getting trapped in this loop, there may be larger or smaller values.)

And from Cpt. DeFrance Clarke III of Fairborn, Ohio:

I modified the Concentration game found on the Sampler disk so that you match numbers rather than finding two that sum to a given number. My two daughters aged 3 1/2 and 6 now play avidly. The modification is simple:

(1) 330 IF(V(G(0))=V(G(1)) THEN 400
(2) 2030 PLOT 6,2,3,15,30:PRINT "FIND MATCHING NUMBERS (0 TO "T")"
(3) 2520 FOR I=0 TO INT(SQ/2):J=FNR(T+1)-L:V(R(I))=J:V(R(SQ-I))=J:NEXT
(4) 11000 PLOT 15,3,0,6,27,24
(5) Either modify the playing instructions (lines 11010-11220) or 11005 RETURN

=

attn/break

ANNOUNCING THE NEW, UPDATED TAX PROGRAM FOR 1979. Compucolor's Income Tax 1979 makes figuring out your 1979 1040 Federal Income Tax form a little less painful. The program requires a 16K machine and retails at the same price as last years's tax program, $29.95 -- no inflation here! Your Compucolor dealer will be happy to take your order now for Taxes 1979.

=

Here's a little something I ran across and decided to reprint for your amusement.

"A computer programmer is one who passes himself off as an exacting expert on the basis of being able to turn out, after innumerable debugs, an infinite series of incomprehensible answers calculated with micro-metric precision from vague assumptions based on debatable figures taken from inconclusive documents of problematical accuracy by persons of dubious reliability and questionable mentality for the purpose of annoying and confounding a hopelessly defenseless department that was unfortunate enough to have asked for the information in the first place."

Now, honestly, how many of you does this refer to?

=

Compucolor® Corporation
Intecolor Drive
225 Technology Park/Atlanta
Norcross, Georgia 30092
Telephone 404/449-5996